REPORT

Ganesh K. (01FB15ECS104)

Abstract:

To implement a B Tree on disk based on Algorithm provided in Introduction to Algorithms by CLRS.

Objectives

The following problems to be solved for given the dataset:

Implement with one of the options for each policy:

1. Insertion and deletion policy
    - Logical deletion: mark the node as deleted get the new node for insertion: from the end of the array note that the deleted nodes become garbage.
    - Keep a free list of nodes get the new node for insertion from the free list on deletion, add the node to the free list.
2. Fixed and varied sized records policy
    - Records could be fixed size or variable size. If the records are of variable size, you require one more level of indirection.
3. Key policy
    - Key could be a single field, could be a composite field (combination of fields).
    - Key could be of any type.
    - Key comparison(predicate) could be flexible.
4. # of passes
    - You may implement a single pass or a two pass policy for insertion and deletion.

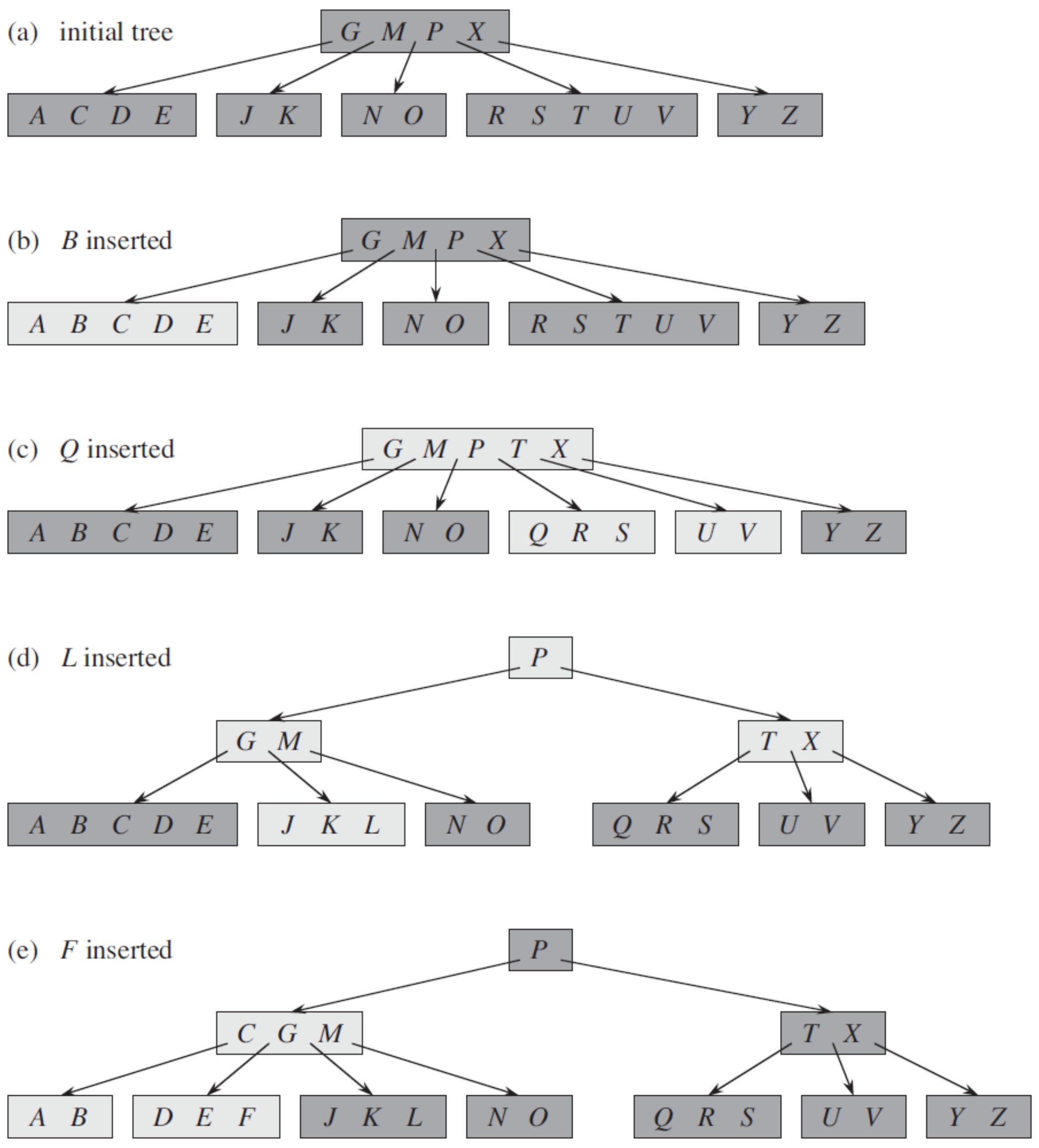
Approach:

The following steps were followed to search the tree:

Source: CLRS:

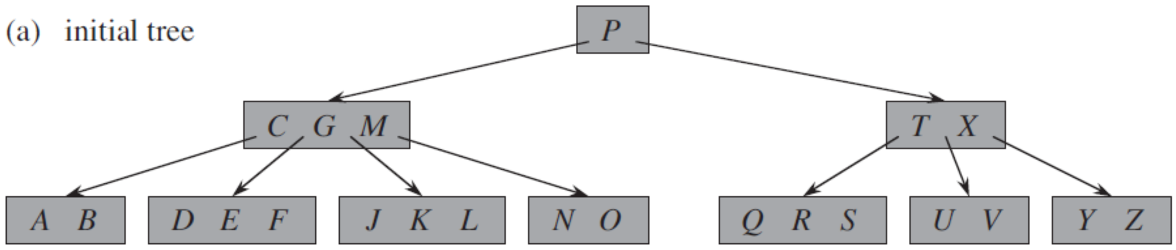
```
B-TREE-SEARCH($x, k$)
$i = 1$
while $i \le x.n$ and $k > x.key_i$
    $i = i + 1$
if $i \le x.n$ and $k == x.key_i$
    return $(x, i)$
elseif $x.leaf$
    return NIL
else DISK-READ($x.c_i$)
    return B-TREE-SEARCH($x.c_i, k$)
```

The following steps were followed to insert into the tree:

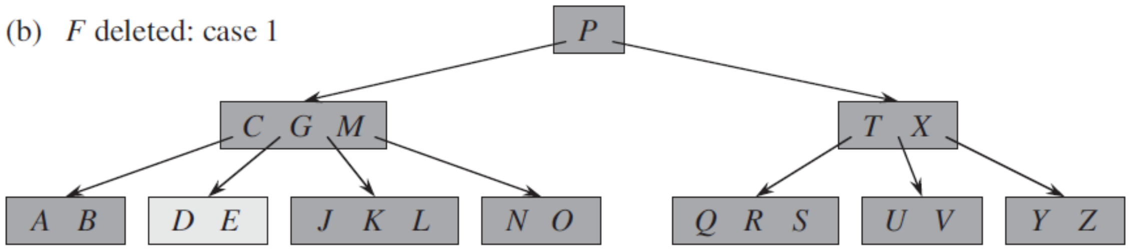


The following steps were followed to delete from the tree:

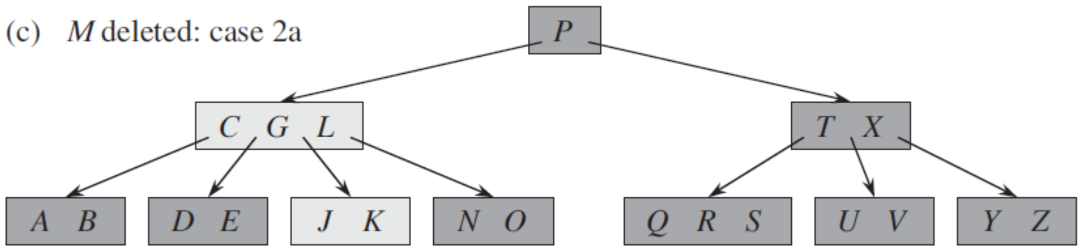
(a) initial tree



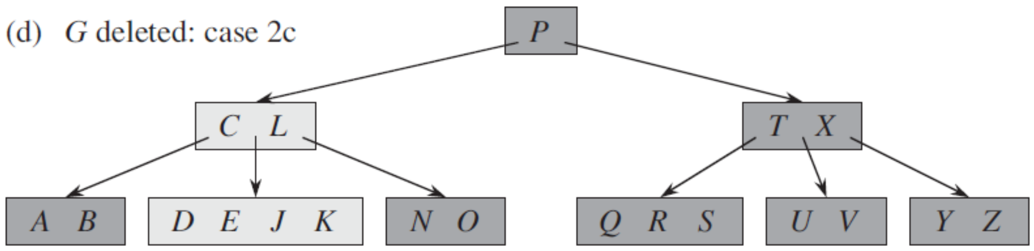
(b) *F* deleted: case 1



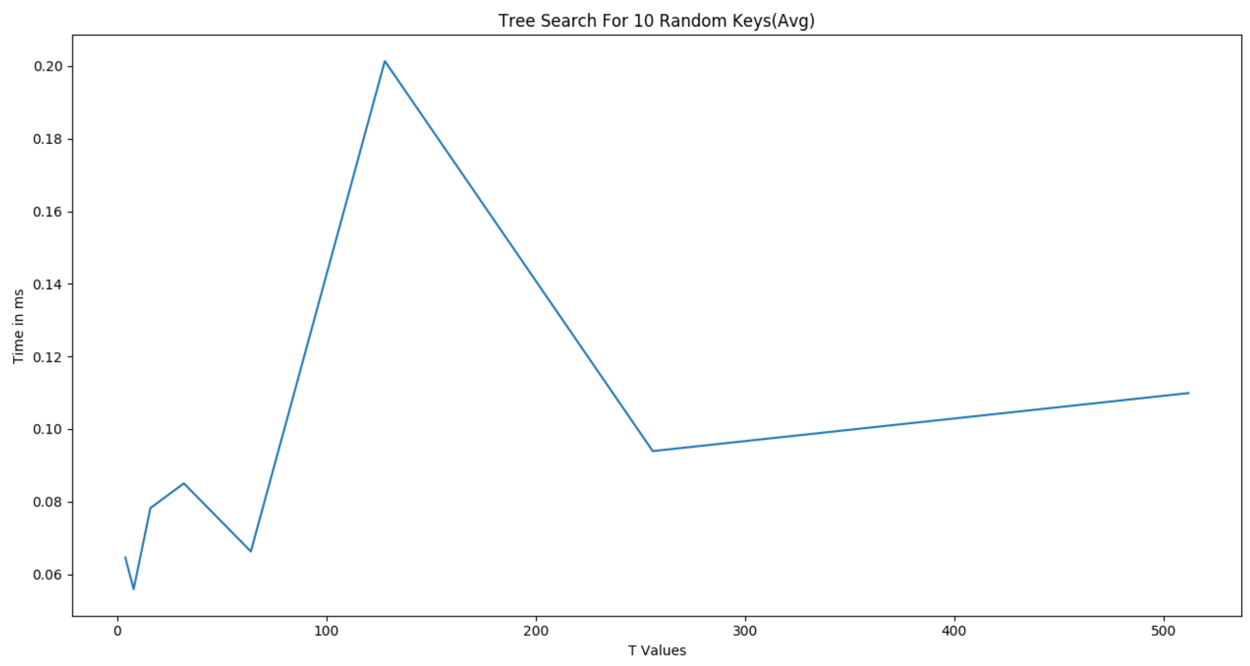
(c) *M* deleted: case 2a

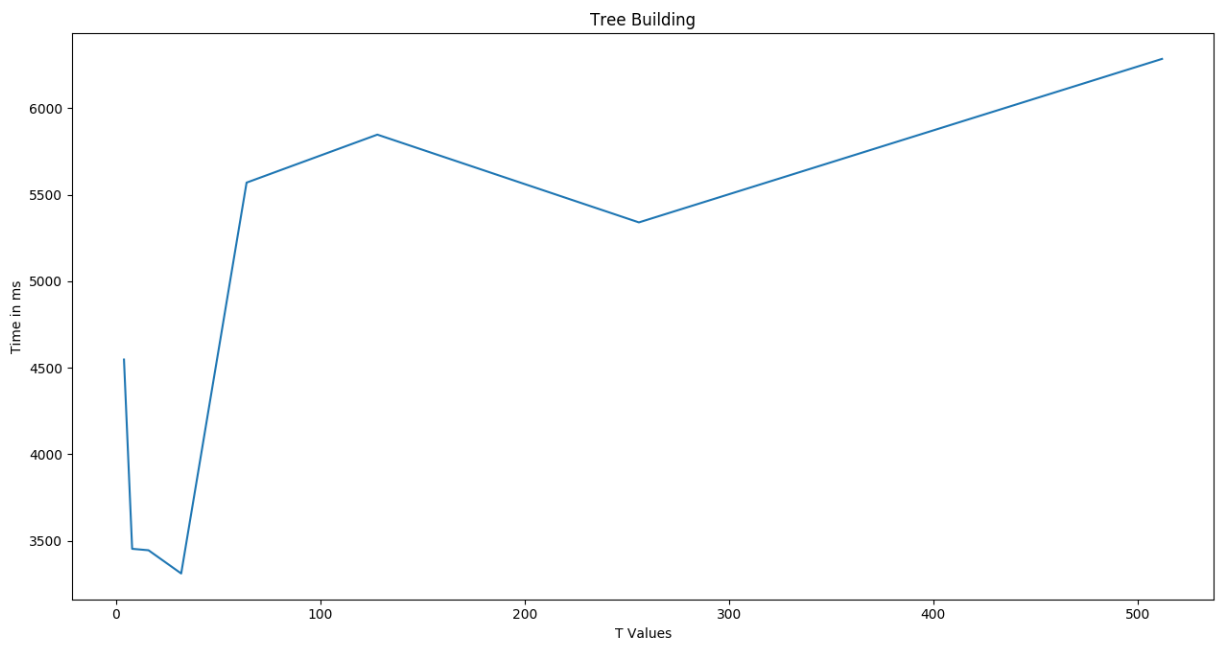


(d) *G* deleted: case 2c



Results:





Result Analysis:

From the above graphs, it is clear that a value of 32 for t is ideal. Below are the possible reasons for this result:

The goal of a b-tree is to minimize the number of disk accesses. If the file system cluster size is 4k, then the ideal size for the nodes is 4k. Also, the nodes should be properly aligned. A misaligned node will cause two clusters to be read, reducing performance.

With a log based storage scheme, choosing a 4k node size is probably the worst choice unless gaps are created in the log to improve alignment. Otherwise, 99.98% of the time one node is stored on two clusters. With a 2k node size, the odds of this happening are just under 50%. However, there's a problem with a small node size: average height of the b-tree is increased and the time spent reading a disk cluster is not fully utilized.

Source: https://stackoverflow.com/questions/2678559/b-tree-node-sizing

Now following are the configurations of my system and the source code for t = 32:

- Page size: 4096 Bytes.
- Node size: 2032 Bytes. Hence Node size is roughly 2K bytes there by satisfying the properties mentioned in the SO

Average Timing: (for t=32)

Problem	Average Time with print (ms)
Insert	0.08504
Build	1074.89

Acknowledgement:

- This is developed as an assignment for Advanced Algorithms Course.
- I would like to thank my professors, Prof. NS Kumar and Prof. Channa Bankapur for their valuable advice.
- I would like to give credit to Prof. NS Kumar for providing necessary code for file handling.
- The delete functions are based on GeeksforGeeks implementation in C++.